UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/535,573	03/27/2000	Robert A. Foster	M-4540-1C us	3655

24251 7590 11/04/2003

SKJERVEN MORRILL LLP
25 METRO DRIVE
SUITE 700
SAN JOSE, CA 95110

EXAMINER	
NGUYEN, CUONG H	

ART UNIT	PAPER NUMBER
3625	

DATE MAILED: 11/04/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary

Application No.	Applicant(s)
09/535,573	Robert A. Foster
Examiner	**Art Unit**
Cuong H. Nguyen	3625

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136 (a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1) ☒ Responsive to communication(s) filed on *Apr 9, 2002* .

2a) ☒ This action is **FINAL.** 2b) ☐ This action is non-final.

3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11; 453 O.G. 213.

Disposition of Claims

4) ☒ Claim(s) *1-86* is/are pending in the application.

4a) Of the above, claim(s) *1-46* is/are withdrawn from consideration.

5) ☐ Claim(s) _____ is/are allowed.

6) ☒ Claim(s) *47-86* is/are rejected.

7) ☐ Claim(s) _____ is/are objected to.

8) ☐ Claims _____ are subject to restriction and/or election requirement.

Application Papers

9) ☐ The specification is objected to by the Examiner.

10) ☐ The drawing(s) filed on _____ is/are objected to by the Examiner.

11) ☐ The proposed drawing correction filed on _____ is: a) ☐ approved b) ☐ disapproved.

12) ☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. § 119

13) ☐ Acknowledgement is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d).

a) ☐ All b) ☐ Some* c) ☐ None of:

1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

*See the attached detailed Office action for a list of the certified copies not received.

14) ☒ Acknowledgement is made of a claim for domestic priority under 35 U.S.C. § 119(e).

Attachment(s)

15) ☐ Notice of References Cited (PTO-892)

16) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

17) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s). _____

18) ☒ Interview Summary (PTO-413) Paper No(s). 19

19) ☐ Notice of Informal Patent Application (PTO-152)

20) ☐ Other:

DETAILED ACTION

1. This Office Action is the answer to the interview on 10/11/2002; this case was transferred to a new address, and is currently handled by Attorney Edward C. Kwok, Reg.# 33,938, at (408) 408-392-9250 X 208.

2. Claims **47-86** are pending in this application.

Response to Amendment

3. Applicant's arguments received on 4/09/2002 have been fully considered but they are not persuasive with previous cited references. The Appeal Conference suggested that the previous Final Office Action was not presented in an organized and clear manner with the order of rejected claims (e.g., some heading paragraphs for rejections were missing); therefore, this Office Action would try to fix that deficiency. This Office Action is a Final Office Action because all the ground of rejections are provided with evidences of Burt et al., Moore et al., Claus et al., and Rothstein et al. that provided to the applicants previously; the examiner apologies for this delay.

4. In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there are some motivations to do so found either in the references themselves **or** in the knowledge generally available to one of ordinary skill in the art. See *In re* ***Fine***, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re* ***Jones***, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, **Moore** et al., **Burt** et al., and **Rothstein** et al. teach applications using object-oriented program (OOP) wherein "instance" are

variable instances (please note that "**instance**" is a macro instruction (in OO programming language, it is a "**reserved term**" that defines a set of instructions that are substituted for the macro name wherever the name appears in a program. Macros are similar to functions in that they can take argument - an instance is a single example or occurrence of a class; e.g., programming instructions to show **Class** vs. **Instance** (from http://scv.bu.edu/Doc/Java/tutorial/java/anatomy/static.html "The Anatomy of a Java Application)

```
Import java.util.Date;

Class DateApp {
        Public static void main(String args[1])  {
                Date today = new Date();
                System.out.println(today);
        }
}
```

In above example, a Class is a Date, and an instance is Monday.

Burt et al. teach a method with related functions including financial transaction functions utilizing instance, although not expressly disclosed claimed phrase of: "creating related records to allow pricing transaction" (see **Burt** et al., Fig.5 – please note that "to allow pricing transaction" is a phrase for intend of use, that phrase is merely an exemplary situation among many different applications of "instance" in OOP); therefore, **Burt** et al. teach analogous functions as claimed (i.e. defining names for a specific application which does not change a required function in programs). **Moore** et al. further clarify such application with rule-based application structure could be a relational database where records of a transaction are related to each other (see **Moore**, the abstract, Figs.3,4, 10:5-34, 23:27-61, these para. indicate relationships of an object and access type with the value in the object instance entity). The examiner

submits that "production service instance" and "billing service instance" in claims merely are examples of financial services OOP software using instance as a macro (e.g., banks charge a service fee for an extra monthly statement from ATMs, banks charge a service fee for transferring cash to a foreign bank due to a request of an account holder; these are financial services that have "linking" with each other by the use of a relational database management system disclosed by **Moore et al**., (please note that "instance" terms in claims are defined as standard situations in an object-oriented program. In Object-Oriented Program (OOP), an **object** is something that has an identity, a state, and a behaviour. the behaviour is encoded in methods (member functions). Objects are bundles of **related variables** and methods and are often used to model real-world objects. It could be said that a class is a blueprint, and an object is a house. An object belonging to a class is referred to as an instance of the class. If humanity were a class, then [you] would be an instance of the class [human]. (Instance variable: In object-oriented programming, an **instance variable** or **data member** is data which an object keeps track of. The examiner submits that in cited references there is a relationship amongst variables; those relationships are called "linkings". Therefore, claimed limitations are obvious with interpretations.(in other words, instances are variables in an object-oriented program, and they are linked directly or indirectly; this is a function/ability of an object-oriented program).

5. The applicant argues that **Burt** et al. fail to disclose "said production service instance being linked to said transaction instance by a first relation instance", and "said billing service instance being linked to said first production service instance by a second relation instance" as recited in claim 47. The term of "being linked" is taught by

Burt et al. as using "connection layer" and "connection instance" (see **Burt** et al., Fig.5),. The examiner submits that Burt et al. teach a relational model as claimed (see **Burt** et al., Fig.5); "relational model is a data model in which the data is organized in relations (tables). This is the model implemented in most modern database management system"; therefore, banking transactions and related pricing were known to implement this model for the pending claimed relational structures (such use of a relational database management software have been applied for banking transactions, see **Moore et al.**, the abstract, Figs.3,4, 10:5-34, 23:27-61 e.g., these para. indicate relationships of an object and access type with the value in the object instance entity).

6. The examiner also submits that suggestions for "Categorization of purchased items for each transaction by a smart card" had been discussed by **Claus** et al. in their patent (see **Claus**, claim 1, and 2:15 to 3:22), and a relational database of items/instances for different transactions were done in a financial software program.

7. The Microsoft Computer Dictionary (published in 1996) defines a standardized meaning of a database wherein data components are linked together within that database as followings: linked list: In programming, a list of elements of a data structure connected by pointers. A singly linked list has one pointer in doubly linked list has two pointers in each node pointing to the next and previous nodes. In a circular list, the first and last nodes of the list are linked together; and link: To produce an executable program from compiled modules (programs, routines, or libraries) by merging the object code (assembly language object code, executable machine code) of the program and resolving interconnecting references (such as a library routine called by a program), or to connect two elements in a data structure by using index variables (index: A listing of keywords and associated data that point to the location of more comprehensive information, such as files and records on a disk/record keys in a

database), or pointer variables (pointer: In programming and information processing, a variable that contains the memory location (address) of some data rather than data itself). The act of linking data/items from different parts in a database is in cited references of **Moore** et al., **Burt** et al., **Rothstein**, **Clause** et al., and they are a fundamental knowledge in database structure of OOPs; from that available computer programming knowledge the applicant uses it to apply for a specific use (i.e. for pricing transactions). Therefore, claim **47** does not teach any new inventive concept according to cited references.

8. The examiner submits that claims **47**, and **68** comprise elements of a relational database structure, would utilize an instance variable in its object-oriented program (i.e., an **instance** is an *instantiated* object of a particular class), an *object* is something that can have properties and relations).

9. At the end, on pg.5, 1st para., the applicant argues that cited references do not teach a "client instance" and a "market segment instance".

The examiner submits that one with ordinary skill in the art would understand that in an object-oriented program (e.g., Java, Visual Basic, C++ .etc.):
- an (entity) instance could be a client instance; an entity instance could be a market segment instance because in OOP, "instance" is a variable: an **instance** is an *instantiated* object of a particular class.

The examiner submits that cited prior art's limitations are not necessary spelled-out exactly claimed languages; analogous interpretations based on definition for

functions of those terms show that such claimed languages would be obvious for meaningful modifications in OOP using in cited art's situations.

10. All claimed limitations have been known since events for pricing transactions always "link" to related objects in a relational database. As the examiner presents that the claimed subject matter is obvious with one of skills in the art, different "instances" in above claims may be defined according to the use of a particular "instance" in an object-oriented program, in relation to the "class" to which it belongs; in other words, instance variable is just a variable associated with an event/action/instance of a class in OOPs (a **class** is a template for a group of objects an object such as: client, market segment with similar behaviour, and a common inheritance).

Claim Rejections - 35 USC § 112

11. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

12.A. Claim **47** is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

It is directed to a method for pricing transactions, comprising:

- creating a database with different instance (e.g. a transaction instance, a production service instance, a billing service instance); these instances are linked by relation instances.

- The examiner submits that there is a gap in this claim about how to perform pricing a transaction after a database is created. This claim is incomplete because a core step how to price is missing although the applicant claims this action; therefore, 35 USC 112, 2nd para. rejection is applied (content of applicant's specification is not used as

evidence that the scope of the claims is consistent with the subject matter which applicant regards as his invention).

12.B. Claim **68** is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. The applicant omits a critical component of claimed system: i.e., a computer to execute an OOP program to create instances. Without that, there is a gap between structures in the claim.

Claim Objections

13. Claims **48-67** are objected for incorporating a defect from parent claim 47 by dependencies.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C.§103(a), which forms the basis for all obviousness rejections set forth in this Office Action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patent ability shall not be negative by the manner in which the invention was made.

14. Claims **68**, **47-53, 55-66, 69-85** are rejected under 35 U.S.C. §103(a) are rejected under 35 U.S.C. §103(a) as being unpatentable over **Moore** et al. (US Pat. 5,630,127), in view of **Burt** et al. (US Pat. 5,682,482).

A. **Moore** et al. teach that a rule-based application structure could be a relational database where records of a transaction are related/linked to each other (see **Moore**, the abstract, and Figs.3,4). **Moore** et al. teach that: service instances linking to transaction instances; and creating a billing service instance linked to a service instance with relation instance (see **Moore**, "FIG. 4 is an object instance table." 6:54-59 "An

example of this table is shown in FIG. 3. The names or "objects" are shown in the columns "OBJECT" 302, "OBJECT1" 304 and "OBJECT2" 308. These names or "objects" stand for a multitude of particular instances of the data, any of which can be retrieved by specifying the identifiers of the entities listed above which would focus the access on a particular representation value."; 10:5-19, 10:45-55 "An additional feature of the GRMS architecture is the placement of the GRMS processor on the Business Professional's workstation 118 along with the Object Table 300, and the programs defined in the object table 300. Since the object instance table 400 is also present, the Business Professional can change values in the Object Instance table (via GRMS screens and functions) and reprocess the report on the workstation. All object accesses will be satisfied by the Object Instance table function and therefore, the CMIM database 224 is not needed for this "What if" analysis reporting."; in OOP, "instance" is a variable name e.g., service instance, relation instance .etc.).

Although Moore et al. teach about a financial institution, and a single transaction can generate many object instances (see **Moore** et al., 1:21-30, and Detailed Description Text portion (para. 439) "Unique identifier for a GRMS transaction. A single GRMS transaction can generate many object instances"), Moore et al. do not explicitly disclose that financial transaction functions are connected together.

However, **Burt** et al. further disclose a system with related functions including financial transaction functions connecting together (e.g. see **Burt** et al., Fig. 5, the abstract, 4:25-27, and 25:2-16), comprising:

- creating a transaction instance corresponding to a financial transaction (e.g. see **Burt** et al., Fig.5, the abstract, col.6 lines 1-14, and col.21 lines 42-59).

The examiner submits that because **Moore** et al. teach applications using OOP macros wherein "instance" is a variable instance - an instance is a single occurrence of a class -, it would be obvious for the analogous use of macros: "transaction instance", "service instance", and "billing service instance".

B. Independent claim **68** recites a data processing system that comprises a means for creating a transaction event, a means for creating a production/service event, and

the linking between events; this claim has similar limitations as of claim 47. Thus it is also rejected for the same rationales and references as claim 47.

C. Re. to claim **75:** The rationales and references for rejecting claim 68 are incorporated.

Moore et al. teach that an OOP software is used for creating different instances (i.e., a fourth relation instance) that links different instances (e.g., a transaction instance to an entity instance), (see Moore et al. Fig.4, and col.10 lines 25-55).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine specific applications to combine Moore et al., Burt et al., in financial transaction with OO programming (for different applications using relational database) because they all suggest a systematic method that use "instance" in a structural database to track all of the components of costs and fees each time a financial transaction is processed. It has been recognized that a finance system would be able to measure profitability in a flexible manner and to measure the impact of any changes from banking clients by tracking those variables.

D. Ref. to claims **76, 77, 56, 57**: The rationales and references for rejecting claim 68 are incorporated.

Burt et al. further teach about storing/retrieving relation instances in relation instance table (e.g., see **Burt** et al., claim 5 - this claim indicates that different rules for objects/instances are stored in tables, and can be retrieved from those tables in OOP); and creating a second entity instance related to first entity instance (e.g. see **Burt** et al., Fig. 4 – this figure indicates that different instances have relationships in OOP).

E. Re. to claims **79, 59, 83, 63, 65, 84**:

The rationales for rejection of claims 68 are incorporated herein.

Moore et al. and Burt et al. also teach :

- an OOP for creating an entity instance relating to another instance (e.g., a transaction instance);
- an OOP for creating an entity instance relating to above entity instance;

Moore et al. also teach a means for creating a price table instance related to an entity instance (see Moore et al., Fig.4); the examiner submits that in OOP variable instances can be created, and can be defined to relate to each other. (The claimed phrase of "wherein a price table instance contains a price for a billing service instance" is a specific but fundamental application of instance variables in OOP).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine specific applications of Moore et al., and Burt et al., in OOP financial transaction (using relational database) because they all suggest a systematic method that use "instance" in an OOP to track components of costs and fees each time a financial transaction is processed. Artisan would recognize that a finance system would be a flexible application to measure the impact of any changes from financial transactions by tracking those instance variables.

F. Re. to claims **60**, & **80**: The examiner's position is that in an OOP software, it is obvious to define that "price table instance is a cost table instance, and a price is a cost" since it is merely defined as a variable instance in Moore et al.'s transaction.

The rationales and references for rejecting claim 79 are incorporated.

The examiner submits that a price table instance could be defined as a cost table instance, and said price could be a cost; or a price table instance could be defined as a fee table instance since price/fee table instance is just a sample instance data structure.

G. Re. to Claims **81, 61:**

The rationales and references for rejecting claim 79 are incorporated.

- The examiner submits that one of ordinary skill in the art would recognize that a price is understood as a fee, whether it is expressed in different term.

H. Re. to Claims **82, 62:**

The rationales and references for rejecting claim 81 are incorporated.

Moore et al., and **Burt** et al. use OOP for their applications. Their programs teach relations/linking between variable instances. Because their instances are variable, it is obvious to name them meaningfully to each specific application.

The uses of a relational database in cited prior art teach a step of creating a cost table instance related to a fee table instance by a relation instance.
Although **Moore** et al., and **Burt** et al. do not specifically disclose that "a cost table instance related to a fee table instance by a relation instance", the examiner submits that any OOP application having a characteristic of linking/relating between object instances. (e.g., see the admission from contents of claims 66, 85 wherein "entity instance" is a variable).

I. Re. To claim **47**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 68. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

15. Re. To Claims **64, 66, 85:**

The rationales and references for rejecting claim 84 are incorporated.

The examiner submits that because an instance is defined as a variable in OOP, an entity instance can be defined as an account instance, a client instance, or can be defined as a market segment instance (see **Burt**, the abstract, claims 1-2).

By contents of the pending claims **66, 85**, the applicant admits that an entity instance is a variable instance, since an entity instant can be used as an account instant or as a client instant.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine specific applications of Moore et al., and Burt et al., in OOP financial transaction because they all suggest a systematic method that use "instance" in an OOP to track components of costs and fees each time a financial transaction is processed. Artisan would recognize that an instant in OOP would be a variable to measure the impact of any changes from financial transactions by tracking those instance variables.

16. Re. To claims **48-53, 58, 69-75, 78, 83**:

The rationales and references for rejecting claim 68 are incorporated.

Moore et al. obviously suggest a step of storing a transaction instance/an account instance/a client instance, a production service instance, a settlement service instance, and a billing service instance in an entity instance table, and they are inherently "link"/"relate" together as a functional data structure (e.g. see Moore et al. Fig.4, and col.10 lines 25-55).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine specific applications of Moore et al., and Burt et al., in OOP financial transaction because they all suggest a systematic method that use "instance" in an OOP to track components of costs and fees each time a financial transaction is processed. Artisan would recognize that an instant in OOP would be a

variable to measure the impact of any changes from financial transactions by tracking those instance variables.

A. Re. To claim **78**: The rationales and references for rejection of claim 68 are incorporated.

- means for creating a settlement service instance linked to said billing service instance by a third relation instance.

Moore et al. teach that an OOP software is used for created different instances (i.e., a settlement service instance) that link/relate (using a relation instance) with another instance (i.e., a billing service instance), (see Moore et al. Fig.4, and col.10 lines 25-55).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine specific applications to combine Moore et al., Burt et al., in financial transaction with OO programming (for different applications using relational database) because they all suggest a systematic method that use "instance" in a structural database to track all of the components of costs and fees each time a financial transaction is processed. It has been recognized that a finance system would be able to measure profitability in a flexible manner and to measure the impact of any changes from banking clients by tracking those variables.

B. Re. To claim **70**: The rationales and references for rejection of claim 49 are incorporated.

This claim further defines that "means for creating a second billing instance linked to said first production service instance by said 2nd service instance".

Moore et al. teach that an OOP software is used for created different instances that link with another instance (e.g. see Moore et al. Fig.4, and col.10 lines 25-55).

C. Re. To claim **49**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 70. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

D. Re. To claims **69, 71-74**: Theses claims are directed to a system of creating a "billing service instance" that link to a transaction instance by a relation instance. The examiner submits that this is an available function of an OOP used by Moore et al. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

E. Re. To claim **48**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 69. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

F. Re. To claim **50**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 71. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

G. Re. To claim **51**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 72. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

H. Re. To claim **52**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 73. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

I. Re. To claim **53**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 74. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

J. Re. To claim **54**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 75. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

17. Claim **55** is rejected under 35 U.S.C. §103(a) are rejected under 35 U.S.C. §103(a) as being unpatentable over **Moore** et al. (US Pat. 5,630,127), in view of **Burt** et al. (US Pat. 5,682,482), further in view of **Rothstein** (US Pat. 5,636,117).

The rationales and references for rejecting claim 54 are incorporated.

Rothstein further teaches that a market segment instance could be an entity instance (see 2:8-10, 2:54-47, 3:9-12) (e.g., mortgage entities are linked to business models by indices in a program).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine specific applications of Moore et al., and Burt et al., in OOP financial transaction with Rothstein because they all suggest a systematic method that use "instance" in an OOP to track components of costs and fees each time a financial transaction is processed. Artisan would recognize that an instant in OOP would be a variable to measure the impact of any changes from financial transactions by tracking those instance variables.

18. Claims **64, 66, 85**, are rejected under 35 U.S.C. §103(a) are rejected under 35 U.S.C. §103(a) as being unpatentable over **Moore** et al. (US Pat. 5,630,127), in view of **Claus** et al. (US Pat. 5,559,313), in view of **Burt** et al. (US Pat. 5,682,482).

The rationales and references for rejecting claim 84 are incorporated.

Claus et al., further express analogous instances in a database, the examiner submits that since they are considered as variable instances in OOPs (see Figs.6, 9-11, 13, 15) for analogous examples that were claimed about:

- an entity instance could be an account instance;
- an entity instance could be a client instance;
- an entity instance could be a market segment instance.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine specific applications to combine Moore et al., Burt et al., and Claus et al. in financial transaction with OO programming (for different applications using relational database) because they all suggest a systematic method that use "instance" in a structural database to track all of the components of costs and fees each time a financial transaction is processed. It has been recognized that a finance system would be able to measure profitability in a flexible manner and to measure the impact of any changes from banking clients by tracking those variables.

19. Re. To claim **56**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 76. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

20. Re. To claim **57**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 77. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

21. Re. To claim **58**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 78. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

22. Re. To claim **59**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 79. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

23. Re. To claim **60**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 80. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

24. Re. To claim **61**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 81. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

25. To claim **62**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 82. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

26. Re. to claim **63**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 83. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

27. Re. To claim **65**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 84. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

28. Re. To claim **66**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 85. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

29. Re. To claim **86**: The rationales and references for rejecting claim 84 are incorporated.

Means for creating a second price table instance related to first entity instance.

Moore et al. teach that an OOP software is used for created different instances (i.e., a second price instance) that link with another instance (i.e., a first entity instance), (see Moore et al. Fig.4, and col.10 lines 25-55).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine specific applications to combine Moore et al., Burt et al., in financial transaction with OO programming (for different applications using relational database) because they all suggest a systematic method that use "instance" in a structural database to track all of the components of costs and fees each time a financial transaction is processed. It has been recognized that a finance system would be able to measure profitability in a flexible manner and to measure the impact of any changes from banking clients by tracking those variables.

30. Re. To claim **67**: This claim is directed to a method of pricing transactions containing similar limitations as in "system" claim 86. Therefore, similar rationales and references set forth are also used for a 35 USC 103(a) rejection.

Conclusion

31. Claims **47-86** are rejected. Claims **48-67** are also objected.

32. **THIS ACTION IS MADE FINAL**. See MPEP 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within

TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

33. The examiner submits that the reasons for rejection are obvious vs. cited prior arts. Applicant is suggested to indicate **in the claims how** the claims distinguish from the combining of cited prior arts. An instance, as defined, is a familiar component in object-oriented programming in relation to the class to which it belongs; a definition for "instance variable": a variable associated with an instance of a class (an object), if a class defines a certain variable, each instance of the class has its own copy of that variable. Hence, there is nothing inventive in defining/creating different instances that linking together in a data structure (the definition is already established for an obvious use of "instance" in cited prior art, i.e. an instance is a single example or occurrence of a class).

34. These references are also considered having similar subject matters to this application:

- **Durand** et al., (US Pat. 5,694,598) teach that an account instance is represented as a data list similarly as an entity instance; a market segment instant could be an entity instance (e.g. see '**117** col.2 lines 8-10, and lines 54-57, col.3 lines 9-12); for a use of instance in OOP in a relational database, wherein different programming instances can be linked together); their patent discloses: "The relational database model was introduced in the early 1970's by

E. F. Codd. Since then, the relational model has become the model employed by most commercial database management systems (DBMS). Data in a relational database is represented as a collection of relations. Each relation can be thought of as a table. Like the relational database model, ***object-oriented*** programming ("OOP") has also existed since the early 1970's. In the early 1990's, ***object-oriented*** programming gained widespread acceptance due to increased power of workstations, proliferation of graphical user-interfaces and the development of hybrid ***object-oriented*** languages such as C++. The OOP paradigm provides a class construct which combines data and procedural abstractions. The definition of a class includes a definition of the storage requirements of the class as well as the procedures which define how objects of the class behave. An object is an **instance** of a class. Every object includes the data and procedural characteristics of its class. In addition, new .objects inherit the storage and functionality defined by all classes used to define the parent of the object. The present proliferation of relational DBMSs coupled with the increasing popularity of the OOP paradigm has resulted in a desire to map data between data models. In particular, it is desirable to access relational databases in OOP applications, and to access ***object-oriented*** data from within a relational DBMS.

Commercial tools currently available for mapping ***object-oriented*** data to relational DBMSs include Persistence, ROCK Phase II, and ObjectStore. These tools are primarily intended to allow application objects to be persistent. Further, these applications typically assume a straight mapping correspondence between application objects and a database schema. Various approaches have been considered for object-relational integration. In most approaches, the purpose has been to interface ***object-oriented*** applications with relational data storage.

These approaches include:

The embedded database interaction in which the interaction is controlled directly by the methods of the object class (e.g. using embedded SQL). This approach makes the ***object-oriented*** application rather tightly coupled to the data-storage technology. It is well suited to code generation techniques when the mapping is straightforward. Persistence is one commercial product incorporating this approach. The import-export approach uses an external module which is invoked as a conversion facility for objects. This approach has been used for conversions between relational and object databases. It can be used for providing persistence and object views to ***object-oriented*** applications. The import-export module acts as an external object server. Although the functional coupling is loose, the module requires information regarding the models on each end and must maintain the consistency of its representations."

- Motivations: "Flexibility: The solution should provide independence from the storage technology;

Composability: The mapping operations and operators should be easy to combine, since requests may concern aggregations of objects such as collections and compositions hierarchies;

Security: The solution should prevent the application designer from accessing the database in an unauthorized or inefficient way;

Evolution: The mapping technique should be flexible with regard to changes in the domain ***object-oriented*** model, in the database schema and in the physical organization; and

"Design overhead": The mapping solution should limit the complexity added to the application object model.".

- **Bohrer** et al., (US Pat. 5,943,497 - August 24, 1999 - 717/121,108), Object-oriented apparatus and method for controlling configuration of object creation, wherein an object-oriented mechanism is disclosed that allows new configuration data to replace existing configuration data within an existing object-oriented program. The new configuration data allows an OO program to be quickly and easily generated from an existing OO program without manually changing the source code of the existing OO program. When a factory object creates an object in the existing OO program, the new configuration data that defines a modified class is used to create the object. In this manner class substitution or redefinition in an OO program is easily accomplished, thereby allowing a new OO program to be easily generated from an existing OO program (or framework).

- **Gudmundson** et al. (US Pat. 5,680,619), applying OOP see Table V; ("Both Elements and Behaviors are "object containers"--in this embodiment, object instances that can "contain" (i.e., be linked to) other object instances. Elements can contain Modifiers as well as other Elements; and Behaviors can contain Modifiers, including other Behaviors").
- **Togawa**, (US Pat. 6,182,156 – 709/316 -1/30/2001), Object-oriented data processing system allowing dynamic change of one or more ancestors and/or descendents of one or more classes of objects, wherein A data processing system at least a portion of which operates under control of an object-oriented program. The data processing system contains an object-oriented system, in which a plurality of objects are defined by the above object-oriented program, each of the plurality of objects belongs to a corresponding one of a plurality of classes as an instance of the

corresponding one of the plurality of classes, and one or more inheritance relationships are defined between at least one pair of classes among the plurality of classes based on inheritance coupling information. An inheritance coupling information setting unit receives a first request, and sets at least one piece of the inheritance coupling information in a data storage area, in response to the first request. The inheritance coupling information held in the inheritance coupling information indicating unit can be referred to by at least one of the above plurality of objects.

- Vic **Arnold** et al., US-PAT-NO: 5936860 – 8/10/1999 – 700/95, Object oriented technology framework for warehouse control, (see 14:31-40, 15:4-22), wherein the patent teaches that a state is encoded in instance variables (data members in an OOP); banking transactions and related pricing were known to implement this model for the pending claimed relational structures (such use of a relational database management software have been applied for banking transactions, e.g., these para. indicate relationships of an object and access type with the value in the object instance entity); (see the abstract, 13:56 to 14:5); and an **instance variable** or **data member** is data which an object keeps track of (see 16:57-65).

- Abstract for WIPO Patent No. 94/20912, Sep. 15, 1994, "Object-Oriented System for Managing Financial Instruments".

- Kauler, "Object-Oriented Flow Design," Dr. Dobb's Journal, Oct. 1996, pp. 54-68.

- E. Cusack, "Inheritance In Object Oriented Z," 1991 European Conf. on Object-Oriented Programming, pp. 167-179, Jul. 1991.

- E. Siepmann and A.R. Newton, "TOBAC: A Test Case Browser for Testing Object-Oriented Software," 1994 Int'l. Symp. on Software Testing and Analysis, pp. 154-168,

Dec. 1994.

- Inspec Abstract No. C9501-7330-007, Salminen et al., 1994, "Modeling Trees Using an Object-Oriented Scheme". Inspec Abstract No. C9412-6110B-221, Berghel et al., 1992, "A Generic Object-Oriented Concurrency Mechanism for Extensibility and Reuse of Synchronization Components".

- Inspec Abstract No. B9409-6210M-025, from Hellemans et al., 1994, "An Object-Oriented Approach to Dynamic Service Descriptions".
Inspec Abstract No. C9409-6180-059, from Wang et al., 1993, "A Framework for User Customization".

- Abstract for WIPO Patent Application No. 94/19740, Goldsmith et al., Sep. 1, 1994 "Framework Processor of Object-Oriented Application".

- Abstract for WIPO Patent Application No. 94/15286, Goldsmith et al., Jul. 7, 1994, "Object-Oriented Framework for Object Operating System".

- Abstract for WIPO Patent Application No. 94/15282, Anderson et al., Jul. 7, 1994, "Dialog System Object-Oriented System Software Platform".

- Abstract for WIPO Patent Application No. 94/15281, Anderson et al., Jul. 7, 1994, "Atomic Command Object-Oriented System Software Platform".

- Abstract from WIPO Patent Application No. WO 9415285, Jul. 7, 1994, "Object-Oriented Notification Framework System", D.R. Anderson et al.

- Abstract for U.S. Patent No. 5,119,475, Schoen et al., Jun. 2, 1992, "Object-Oriented Framework for Menu Definition".

- Allard et al., Text of IBM Technical Disclosure Bulletin, Feb. 1990, "Object-Oriented Programming in C--the Linnaeus System", pp. 437-439.

- J. Knapman, Text of IBM Technical Disclosure Bulletin, vol. 38, No. 1, Jan. 1995, pp. 411-414, "Generating Specific Server Programs in Distributed Object-Oriented Customer Information Control System".

- Imamura (US Pat. 5,560,014 with priority date 6/02/1994), Dynamic object management method in object oriented languages.

- **Gudmundson** et al. (US Pat. 5,680,619 filed on 4/03/1995) about a hierarchical encapsulation of instantiated objects in a multimedia authorizing system.

- **Durand** et al. (US Pat. 5,694,598 filed on 10/12/1994) about a method for mapping data between a relational format and an object-oriented format.

- Brow et al., Billing method for goods and service, involves downloading user selected item and incorporating into associated page, during which item dependent billing event is generated, wherein this patent discusses that a server generates an item-dependent billing event at the time of loading the item; priority date: 08/18/2000 (from Dialog® File 350, acc. No. 2002-434636/200246).

- **Hoffman**, Billing system associate predefined pricing rule to each customer corresponding to service utilized by customer, wherein the inventor suggests that an association module associate a predefined pricing rule to each customer corresponding to the service utilized by the customer, and creating unique pricing rules for each customer; priority date: 5/15/1998, (from Dialog® File 350, acc. No. 2002-414149/200244).

- **Wilcox**, Billing method for Internet users, involves determining billing events by extracting operational parameters from data files using software billing rules; priority date: 10/29/1999 (from Dialog® File 350, acc. No. 2001-564848/200163).

- **Boardman** et al., Decision network that rates, prices or discounts transactions based on business rules stored in a price plan selects a particular price plan and an algorithm rule and conditions applicable to certain transactions; wherein price plans and decision networks are stored and, in response to the telecommunications billing event to be priced, a processor traverses a plan selection rule and processing conditions within the plan selection rule to select a price plan applicable to the event. The processor then traverses an algorithm selection rule set of the selected price plan and processing conditions within the algorithm selection rule set to select a pricing algorithm which is then used to price the event; priority date: 07/29/1998 (from Dialog® File 350, acc. No. 2000-223908/20019).

- **Denard** et al., Multi-fuels marketers, Oil & gas investor Performance powered: the new value drivers for the energy supplement, pp:14-17, 2nd quarter 1997; wherein general ideas about pricing schemes for product-service transactions are suggested.

- **Braradwaj** et al., Determinants of success in service industries: a PIMS-based empirical investigation, Journal of Services Marketing, v7n4, pp.19-40, 1993 (from Dialog® File 15, acc. No. 00813287) wherein this article discloses that "The empirical results using the profit impact of market strategy (PIMS) database suggest that integrating forward, having a relatively larger market share, sharing customers with other business units in the firm, and being a market pioneer positively influences

financial position. Customizing the service, integrating forward, sharing customers with other business units in the firm, having a strong service image, and being in a market with a small number of competitors positively influences relative market share. Finally, having a strong service image, improving service quality, and having a higher relative price reduces the risk to service firms.".

- **Stuchfiel** et al., Modeling the profitability of customer relationships: Development and impact of Barclays de Zoete Wedd's BEATRICE, Journal of Management Information Systems: JMIS, v9n2, pp.53-76, Fall 1992 (from Dialog® File 15, acc. No. 00813287) wherein this article discloses that "Activity-based cost (ABC) accounting methods offer a solution, and several firms are developing information system (IS) to gather and process cost and revenue data using these techniques".

- **Price** et al., WO 200235434 A2 – 5/02/2002 with priority date: 10/20/2000), A system providing event pricing for on-line exchange. Although analogous pricing transactions are disclosed (see claims 16-18), the priority date of this reference is after pending application.

- Text of IBM Technical Disclosure Bulletin, **Allard** et al., Feb. 1990, "Object-Oriented Programming in C--the Linnaeus System", pp. 437-439.

- Text of IBM Technical Disclosure Bulletin, vol. 38, No. 1, Jan. 1995, pp. 411-414, J. **Knapman** "Generating Specific Server Programs in Distributed Object-Oriented Customer Information Control System".

- Abstract from U.S. Patent No. **5,371,891**, "Object Constructions in Compiler in Object Oriented Programming Language", J. Gray et

al., Dec. 6, 1994.

- Abstract from EPO Patent Application No. **EP 622730**, "Encapsulation of Extracted Portions of Documents Into Objects", M.A. Malamud, Nov. 2, 1994.

- Abstract for EPO Patent No. 619544, S. **Danforth**, Oct. 12, 1994, "Language-Neutral Object-Oriented Programming".

- Abstract for **WIPO Patent** No. **94/20912**, Sep. 15, 1994, "Object-Oriented System for Managing Financial Instruments".

- Inspec Abstract No. C9504-7460-043, **Sells** et al., 1995, "Implementation of the Architecture for a Time-Domain Dynamical System Simulation in a Very High-Level Pictorial Object-Oriented".

- Inspec Abstract No. C9504-7460-042, **Coleman** et al., 1995, "An End-To-End Simulation of A Surveillance System Employing Architecture Independence, Variable Fidelity Components and Software Reuse".

- Inspec Abstract No. C9503-6140D-045, **Satoh** et al., 1995, "Process Algebra Semantics for a Real Time Object Oriented Programming Language".

- Inspect Abstract No. C9408-6110J-011 from **Gyu-Chung** et al., 1993, "System Methodologies of Object-Oriented Programs".

- Inspec Abstract No. C9407-7420D-045, from **Desai** et al., 1994, "Controller Structure Definition Via Intelligent Process Control".

- Inspec Abstract No. C9407-6140D-014, from **Satoh** et al., 1994, "Semantics for a Real-Time Object-Oriented Programming Language".

- Inspec Abstract No. 4459325, from **Kesim** et al., 1992, "On the Evolution of Objects in a Logic Programming Framework".

- Inspec Abstract No. 4447153, from **Klein** et al., 1992, "An Object-Oriented Framework for Curves and Surfaces".

- Inspec Abstract No. 4426852, from **Benveniste** et al., 1992, "Concurrent Programming Notations in the Object-Oriented Language Arche".

- Inspec Abstract No. 4425343, from **Demurjian** et al., 1993, "Programming Versus Databases in Object-Oriented Paradigm".

- Inspec Abstract No. 4417604, from **Kraiem** et al., 1992, "Mapping of Conceptual Specifications Into Object-Oriented Programs".

- Inspec Abstract No. C91041980, from **Choi** et al., 1991, "Graph Interpretation of Methods: A Unifying Framework for Polymorphism in Object-Oriented Programming".

35. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Cuong H. Nguyen whose telephone number is 703-305-4553 The examiner can normally be reached on Mon.-Fri. from 7:15 AM to 3:15 PM (EST).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ms. Wynn Coggins, can be reached on (703)308-1344.

Any response to this action should be mailed to:

Amendments

Commissioner of Patents and Trademarks
Washington D.C. 20231

or faxed to:

[Official communications; including
After Final communications labeled
"Box AF"]

703-746-5572 (RightFax) Informal/Draft communications, labeled
"PROPOSED" or "DRAFT"]

Hand delivered responses should be brought to Crystal Park 5, 2451 Crystal Drive, Arlington, VA, 7th floor receptionist.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the Receptionist whose telephone number is (703)308-1113.

Cuonghnguyen
Oct. 23, 2002